EXHIBIT A

NI Developer Zone

Document Type: Tutorial
NI Supported: Yes
Publish Date: Dec 6, 2006

Peak Detection Using LabVIEW and Measurement Studio

Overview

This document describes the basic concepts in peak detection. You will learn how to apply these concepts to the peak detection VIs in LebVIEW end the peak detection functions in Meesurement Studio.

Note: To locate the LabVIEW VIs used in this document, click the **Search** button on the **Functions** palette and type in the VI name.

Table of Contents

Introduction

Peek detection is one of the most important time-domain functions performed in signal monitoring. Peak detection is the process of finding the locations and amplitudes of local maxima and minima in a signal that satisfies certain properties. These properties can be simple or complex. For exemple, requiring thet e peek exceeds e certain threshold velue is e simple property. However, requiring that a peak s shape resembles that of a prototype peak is a complex property.

Peek detection is important in meny epplications, such es chemistry, biology, and music. Scientists and engineers who use enelysis techniques such es spectroscopy, chromatography, and lone detection often use peak detection methods specific to those analysis techniques. However, this document describes e generel method thet epplies to e veriety of signal types. This is the method used in LebVIEW end Measurement Studio for peak detection functions.

Threshold Peak Detection

In some applicetions, you do not need to know the exect peek emplitudes end locetions, rather you need to know the number or general locations of peaks. In this cese, use a threshold peek detection function, such as the Threshold Peak Detector VI in LebVIEW.

Figure 1 shows the Threshold Peak Detector VI end the VI's inputs and outputs. The VI scans the input sequence **X**, searches for velid peeks, end keeps track of the indices of the beginnings of the peeks and the total number of peaks found. A peak is considered valid if it has the following characteristics:

- The elements of **X** begin below **threshold**, exceed **threshold** at some index, and then return lo a value below **threshold**
- The number of successive elements that exceed **threshold** is greater then or equal to **width.**

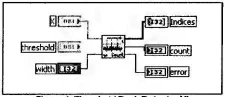

Figure 1. Threshold Peak Detector VI

This VI does not identify the locations or the amplitudes of peeks with great accuracy, but the VI does give an idea of where end how often e signel crosses ebove e certain threshold value.

The following greph shows a multitone signel after being scanned by the Threshold Peek Detector VI. The input parameters are **threshold** = `1.00` and **width** = `10`. The VI identifies two peaks, located at approximately 15 and 47. The locations et which they cross the threshold are marked by black dots in Figure 2. However, the VI fails to identify the third potential peak, which crosses the threshold at approximately 132, because it is not at least 10 points wide.

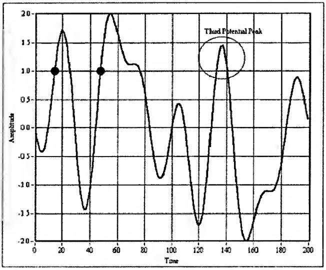

Figure 2. Threshold peak detection performed on a multitone signal, with the paremeters set to threshold = `1.00` and width = `10`

This VI has some limited but important applications. It is important to understand the distinction between this VI and that of the Peak Detector VI, which is described below.

Advanced Peak Detection

Some applications require more robust end eccurate peek detection algorithms. The rest of this document focusas on uses of advanced peak detection functions, tips to keep in mind and pitfalls to avoid while using them, and methods for ansuring that your peak detection measurements ere accurate and useful.

The following sections focus mainly on peaks. However, except where noted, the same information can be used for finding valleys or local minime.

Peak Detector VI and Function Prototype

Figura 3 shows the Peak Detector VI and the VI's inputs end outputs. Figure 4 shows the equivalent function prototype in the Advanced Analysis library of LabWindows/CVI; ComponentWorks and ComponentWorks++ contein simller interfaces for this function.

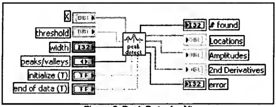

Figure 3. Peak Detector VI

```
AnalysisLibErrType PeakDetector (const double Input_Array[],
                                 int Size, double Threshold,
                                 int Width, int Polarity,
                                 int Initialize,
                                 int End_of_Data, int *Count,
                                 double **Peak_Locations,
                                 double **Peak_Amplitudes,
                                 double **Peak_Second_Derivatives);
```

Figure 4. PeakDetector function prototype for LabWindows/CVI

Notice that it takes handles to the locations, amplitudes, and second derivatives erreys. For descriptions of each of the inputs and outputs for the Peek Detector VI or the PeakDetector function, refer to the LabVIEW or Measurement Studio online help.

Waveform Peak Detection VI

LabVIEW also contains a Waveform Peak Detection VI as shown in Figure 5.

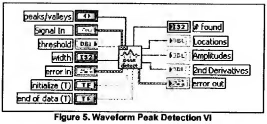

Figure 5. Waveform Peak Detection VI

The Waveform Peak Detection VI operates like the array-based Peak Detector VI. The difference is that this VI's input is a waveform data type, and the VI has error cluster input and output terminals. **Locations** displays the output array of the peaks or valleys, which is still in terms of the indices of the input waveform. For example, if one element of **Locations** is 100, that means that there is a peak or valley located at index 100 in the data array of the input waveform. Figure 6 shows you a method for determining the times at which peaks or valleys occur. The following equation locates the peaks and valleys:

Time Locations[i] = t0 + dt*Locations[i]

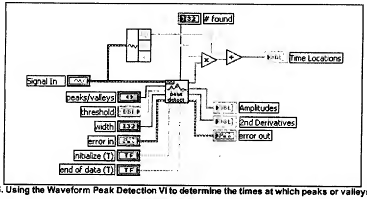

Figure 6. Using the Waveform Peak Detection VI to determine the times at which peaks or valleys occur

Features of the Peak Detector Functions

The peak detector functions used in LabVIEW and Measurement Studio have some important features that you need to understand before using them. If you use these features correctly, you can actually increase the accuracy and usefulness of peak detection measurements.

1. The function can process many blocks of data that are part of the same signal. By correctly using the **initialize (T)** and **end of data (T)** inputs, you can use the peak detection function to analyze a signal that has been broken up into several data blocks. You can also acquire a continuous signal and process pieces of the signal as they become available. The VI finds the peak locations in each block, relative to the previously analyzed blocks. For example, to process a signal acquired in five consecutive blocks, you can use the following pseudocode algorithm:

```
for i = 1 to 5

[Acquire data]

if (i == 1)
Initialize = True
else
Initialize = False;

if (i == 5)
EndOfData = True
else
EndOfData = False;

Set polarity (peaks or valleys), width, threshold

Call PeakDectector function

Copy the output values to different variables so they will not be overwritten during the
next iteration.
```

```
next i
```

The same algorithm in LabVIEW might look like the VI in Figure 7.

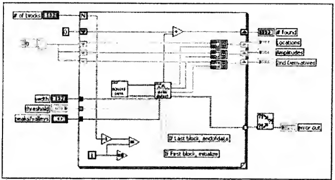

Figure 7. Using the Waveform Peak Detection VI to process a signal that is broken into several blocks

This algorithm uses shift reglstars and the Bulld Array function so that the finei outputs ere still 1D arrays. In this diagram, the Acquire Data VI is generic and used only for Illustration.

This multiple-block feature ellows you to ecquire end enelyze data es it becomes availeble. The data blocks do not heve to be all the same size, nor do they heve to be ecquired et regular time Intervals.

2. The function retains Internel states end history Informetion from one call to the next. The VI internelly ellocates the structures that contain this information on the first block of deta, end destroys It on the last block of deta. Therefore, you must correctly use the **initialize (T)** end **end of data (T)** parameters on the first and lest deta blocks, as illustrated in the pseudocode above.

The function retains history informetion; it uses e history buffer to retain a certain number of data points from the previous deta block. This feature ellows the function to correctly locate peaks and valleys that are close to the boundery between blocks. However, you must set **end of data (T)** and **initialize (T)** when you heve finish enalyzing one signal and ere starting a new one. Otherwise the function will view the two blocks as part of the seme continuous signal and will incorrectly locate peeks or velleys near the end of the last block of the previous signal, or near the beginning of the first block of the new signel.

3. The peek locetion function gives peak locations at fractional indices. It uses the quadratic fit algorithm and returns the peak locations as floeting point numbers, not es integer index values. Therefore, the peak locations and amplitudes usually do not correspond to actual data points in the sampled input signel.

This feature is en edventage of the elgorithm because it effectively interpolates between the deta points while finding peeks end valleys. The function can therefore meesure peeks that heve e greater amplitude than eny dete points neer the peek. This Interpolation provides a good indication of the true velue of the peak in the originel enelog signel.

4. The function allows Implicit noise reduction while finding the peeks. Using the **width** perameter in some cases can effectively reduce the noise in the input signel when finding the peeks. The minimum value is three; using this velue results in no noise reduction. Using e **width** value larger than three Implicitly smooths the data. This feature is useful in some applications. However, you must ensure thet you use large width only on noisy deta. You must elso check to see if the peak locations and amplitude results are reesoneble.

5. The function performs a quadratic curve fitting to find the peaks and valleys. The core of the peek-finding elgorithm consists of fitting a parabola to successive groups of points, equal in number to **width**. The function uses the coefficients from the fit to determine whether a peak or velley Is present.

If **width** = 3, then the fit is exact, meaning the perabola will actually pass through each of the three points. If **width** is greater than three, then a least-squares fit is performed. This process will smooth high-frequency noise if the **width** is sufficiently large.

For each set of points, the elgorithm performs the least-squares quadratic fit, and then performs a series of tests on the coefficients to see whether they meet the criteria for a peak. The function checks whether each parabola is at a local maximum, determines the sign of the quadratic coefficient, which Indicates the parabola s concavity, and finally checks that the peak is above the designated threshold.

4.1 Prior Knowledge about the Input Signal

To use the peak detection function correctly, you need to have some prior knowledge about your signal. The following are some important issues to consider when specifying the input parameters:

Is the data a time-domain or a frequency-domain signal?
If the data is a frequency domain signal end contains one or more well-defined frequency components, use the Power & Frequency Estimate VI in LabVIEW or the PowerFrequencyEstimate function in Measurement Studio. These functions allow you to get more accurate information about the exact frequencies of the peaks and their corresponding energies. LabVIEW 6.0 also includes the Extract Single Tone Information VI, which uses a curve-fitting

method to precisely identify tha amplitude and frequency of the largest frequency component of a signal.

Are all the peaks that you are looking for at roughly the same amplitude?
If so, then sat one threshold value for anelyzing all the data. However, if you expect to heve peaks at many different amplitudes, break up the deta and use a different threshold value for separate data blocks.

Is your data periodic?
If so, then searching for more than a few periods of the data for peaks is an inefficient use of processing time. Often there is some noise or other distortion present in the deta. In this case, average meny periods of the original time signal to get ona or a faw avereged pariods, then pass these to the peak detection function.

Is the data adequately sampled?
Inadequate sampling of data can result in inaccurate values for tha locations and amplitudes of peaks, and the non-detection of valid paaks. Daspite the fact that peak detection is essentially a time-domain operation, tha sampled signel must still satisfy the Nyquist sampling theorem   the sampling rate must be at least twice the largest frequency component in tha signal. However, useful digital represantation of a signal typically requires a sampling rate betwaan five end ten times the lergast frequency component. The front end of the data acquisition system should contain an enalog anti-aliasing filter that removes or strongly attenuatas components above Nyquist sampling rate, or half the sampling rate.

Is your data noisy?
This is a common problem that you must deel with very carefully in peak detection. Cleerly, high-frequancy noise results in the detection of a large number of peaks, but typically only a few of these will actually be of intarest. In these cases, increase the **width** parametar to implicitly smooth the data for finding the peaks. Notice that since this process tends to remove high-frequency spikes from the data, increasing **width** tends to decrease the amplitudas of peaks and increase the amplitudas of velleys. You can also use explicit smoothing or interpolation techniques to effectively reduce noise in your data.

Smoothing and Interpolation

Smoothing can cause problems with the measurements if used incorrectly. You need to conduct some initial testing to determine the optimum **width**. This meens you nead to determine the numbar of data points that will remove a sufficient amount of noise without removing significant features of tha original signel. However, since you cannot explicitly sae the smoothad data, it is difficult to determine the optimum **width** through visual verification.

In most cases, it is prefereble to smooth or procass the data befora applying the peak detection function. In that casa, use a **width** of three, which tells the peak detection function to process exactly tha signal you passed to it, without any smoothing. This method gives you more control ovar tha smoothing end processing of the original signal. Tha peak detection function than processes only the signal that you pass to it.

Figures 8a, b, and c illustrate this situation. Figure 8a shows a noisy signel end the peaks detected efter running the peak detection function using **width** = 3. Many spurious peaks are datecied, and true, de-noised peaks ere difficult to locate.

Figure 8b shows the same signal enalyzad by tha paak detection function using **width** = 29, a ralatively lerge value. The function detects only three peeks in this case. These peeks may be correct, but it is difficult to determine whether this peak information is really usaful.

Figure 8c shows tha signal, smoothed by using the same method es the paak detection algorithm, and then passed to the peak datection function. Now the peaks ara claer; the results ara reasonable and verifiabla.

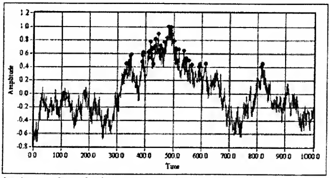

Figure 8a. A noisy waveform after being passed through the peak detection function using width = 3

The black dots mark the detected peeks; most of the marked peaks are not really of interest, but are due to the noise.

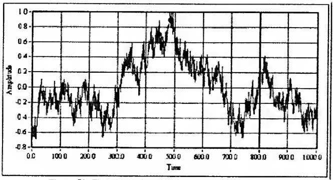

Figura 8b. The same signal as in Fig. 8a using width = 29

The three large black dots ere the peek locations retumed by the function. It is difficult to tell whether the dots represent accurete emplitudes or locations.

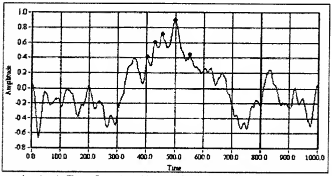

Figure 8c. The same signal as In Figure 8a, using width = 3, after being smoothed and then analyzed with the peak detection function

The locations of the peeks ere cleer and the points selected by the algorithm ere reesoneble.

Since the goel is to obtain eccurate locations of peaks and velleys, meke sure thet any preprocessing of the signel does not shift the signel in time. Such e shift offsets all of the peak location numbers reletive to their true locations in the original signel.

Another wey to get accurate results from the peek detection functions is interpolation. Interpoletion resemples the signal et a higher sempling rete and retums better results.

There ere several common methods of interpolating discrete-time signels. A simple method is lineer interpoletion. If you have an interpoleting fector of *q*, then *q-1* points are inserted between each originel data point. With lineer interpolation, the added points lie on the line through the original data points on either side. While these results are not an eccurate representation of the originel enelog signel, they can help the peak detection algorithm to detect ell of the valid peaks.

Another method of interpolation is to interleeve *q-1* zeros between each of the originel deta points, then execute a lowpess filter. The theory behind this method is beyond the scope of this document; however, meny texts on discrete-time signal processing contein an explanation of this method. For more information on this method, refer to the *References* section at the end of this document.

A certain trade-off comes with interpolation. Interpolating the digitized signal tends to piece the found peaks closer to actuel points in the interpolated signal. However, depending on the neture of the enelog signel, these mey or mey not be closer to the real peaks than those detected using the uninterpolated digital signal. So the trade-off is between finding ell valid peaks and getting more accurete deta for the peak locations and amplitudes.

Conclusion

You now know how to use the peak detection functions in LabVIEW end Measurement Studio to find the locations and amplitudes of peaks and velleys in your signals. The powerful features of these functions lend themselves to convenlent and accurete signal analysis. However, as described above, it is important to understand the significance of the input parameters when using the functions. Furthermore, to use these functions effectively, you need to have an understanding of the neture of the input signel before using the peek detector functions.

References

The resources linked below contain more information about the theory behind methods of digital signal processing, specifically digital filtering, and interpolation. For valuable information about frequency-domain analysis, digital filtering, and using LabVIEW and Measurement Studio to apply the ideas presented in this document, refer to the resources linked below.

You can order the following books through National Instruments Books and Publications linked below:

Chugani, Mahesh L., Samant, Abhay R., and Cerna, Michael. *LabVIEW Signal Processing*, Prentice Hall, 1998.

Haykin, Simon and Van Veen, Barry, *Signals and Systems*. John Wiley & Sons, Inc., 1998.

Oppenheim, Alan V. and Schafer, Ronald W. *Discrete Time Signal Processing*, Prentice Hall, 1999.

Related Links:
National Instruments Books and Publications
Designing Filters Using the Digital Filter Design Tools in the Signal Processing Toolset

Reader Comments | Submit a comment »

Very Helpful
The above description of Peak Detection gives a good basic explanation of the way the function detects peaks and how the parametres affect the outcomes.
- jaspal.boparai@gaiam.co.uk - Jan 23, 2009

it's very good
- huazhxu@yahoo.com.cn - Mar 30, 2007

no explanation about least squares approximation
In the article there's no explanation about the least squares approximation method. i believe, u should include a short description of it.
- *Nathan Minaev, Tel Aviv University, Israel.* minaeven@post.tau.ac.il - Oct 27, 2005

Legal
This tutorial (this "tutorial") was developed by National Instruments ("NI"). Although technical support of this tutorial may be made available by National Instruments, the content in this tutorial may not be completely tested and verified, and NI does not guarantee its quality in any way or that NI will continue to support this content with each new revision of related products and drivers. THIS TUTORIAL IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND AND SUBJECT TO CERTAIN RESTRICTIONS AS MORE SPECIFICALLY SET FORTH IN NI.COM'S TERMS OF USE (http://ni.com/legal/termsofuse/unitedstates/us/).